

AD-A279 929



**Two Synthesis Applications:  
Boeing/Navy/Stars Flight Training  
Systems and Rockwell Command  
& Control Systems**

DTIC

ELECTE

JUN 03 1994

S

F

D

94-16276



SPC

2214 Rock

ndon, Virg

(703)

**Best  
Available  
Copy**

# **TWO SYNTHESIS APPLICATIONS:**

## **BOEING/NAVY/STARS FLIGHT TRAINING SYSTEMS**

**K.C. King**

## **AND ROCKWELL COMMAND & CONTROL SYSTEMS**

**Jerri Turner-Harris**

**SPC-94035-CMC**

**VERSION 01.00.00**

**MAY 1994**

Accession For	
NTIS CRA&I	A 1 1
DTIC TAB	
Unannounced Justification	
By _____	
Distribution / _____	
Availability Codes	
Dist	Avail and/or Special
A-1	

This material is based in part upon work sponsored by the Advanced Research Projects Agency under Grant # MDA972-92-J-1018. The content does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

This document accompanies a videotape of the same presentation recorded live at the Software Productivity Consortium in March 1994. It is recommended that the videotape be viewed with these viewgraphs at hand.

Produced by the  
SOFTWARE PRODUCTIVITY CONSORTIUM  
under contract to the  
VIRGINIA CENTER OF EXCELLENCE  
FOR SOFTWARE REUSE AND TECHNOLOGY TRANSFER  
SPC Building  
2214 Rock Hill Road  
Herndon, Virginia 22070

**94 6 1 001**

# ABSTRACT

## TWO SYNTHESIS APPLICATIONS: BOEING/NAVY/STARS FLIGHT TRAINING SYSTEMS AND ROCKWELL COMMAND & CONTROL SYSTEMS

K.C. King, Boeing STARS Demonstration Project Manager  
Jerri Turner-Harris, Rockwell's Command and Control Systems Division

K.C. King presents the Boeing/Navv/STARS demonstration of the Consortium's Synthesis methodology for software reuse on Navy Flight Training Systems. Synthesis is the foundation for an approach being used by Boeing and a U.S. Navy/STARS team to develop software for the Navy's T-34C flight simulator trainer. A major reuse strategy of the program is the use of a "two life-cycle model", requiring that the traditional application development life cycle be "front-ended" with a separate, but coordinated, life cycle that creates process-driven software assets for a defined product line (or family) of systems. The mastering of variations in requirements between similar systems is at the heart of Synthesis. Mr. King describes how they have used Synthesis in a comprehensive, "leveraged" mode, fully deploying the Synthesis methodology throughout the program.

Jerri Turner-Harris of Rockwell's CCSD discusses their use of Synthesis to create a "domain engineering" approach to developing interprocessor communication and message handling systems for commercial clients. Ms. Turner-Harris explains how Rockwell is implementing Synthesis in their communications domain to eventually allow them to develop new systems in a fraction of the time currently required.

This video is intended for systems and software engineering development lead engineers, project managers, and division managers working in the area of system and software engineering. Viewers will benefit by gaining insight into the synthesis process and the experiences of others who are successfully applying synthesis.

1

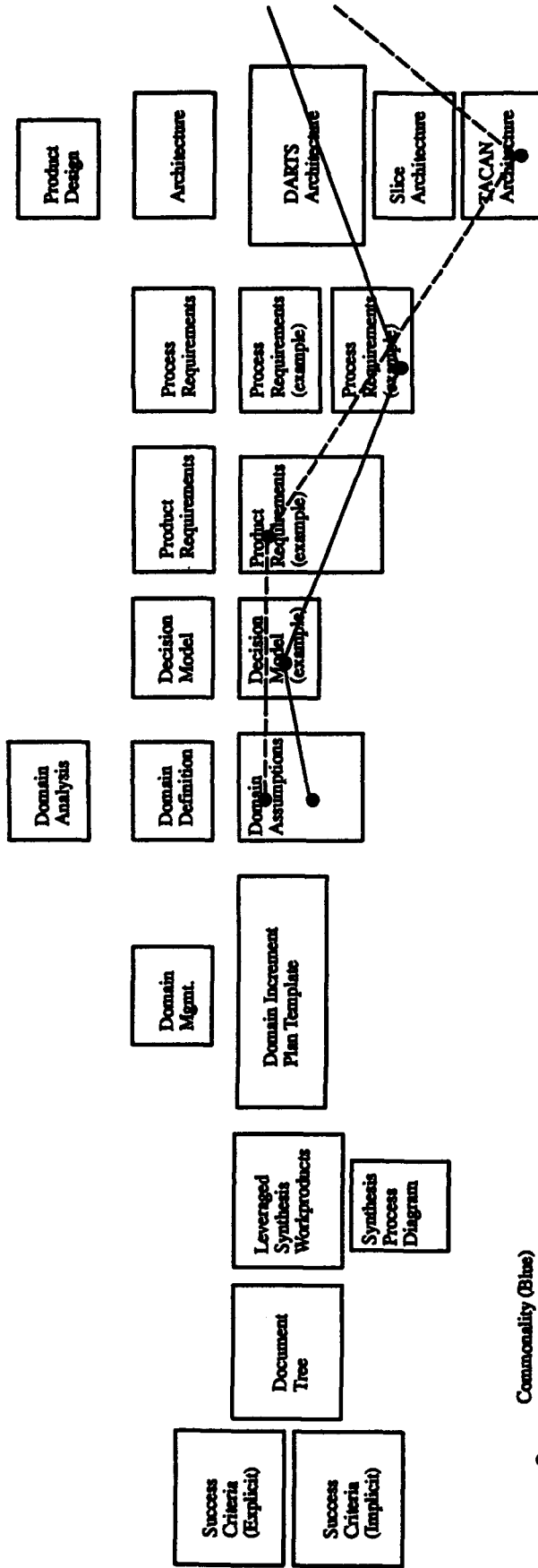
# **ARPA STARS PROGRAM**

## **Boeing/Navy Demonstration Project**

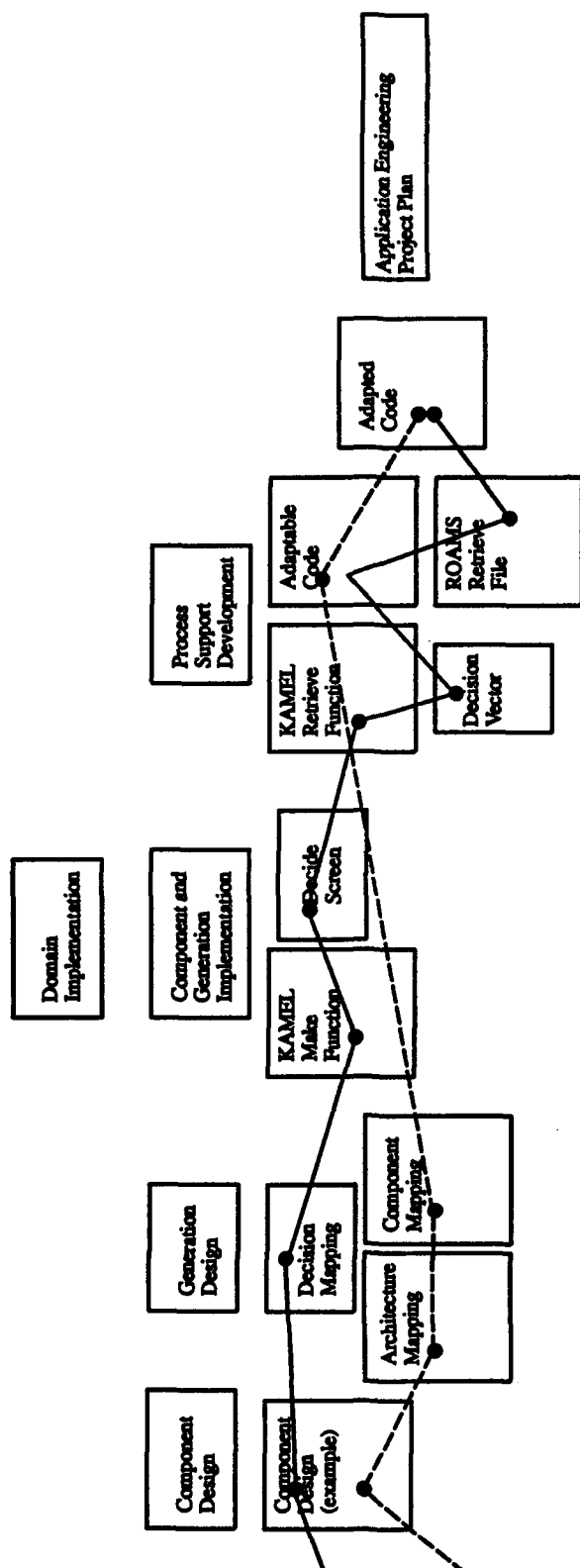
### **Navy Flight Training Systems**

**Presented by**  
**K. C. King**

**Boeing STARS Demo Program Manager**



Overview of "The Wall" (1 of 2)



## Overview of "The Wall" (2 of 2)

## **Success Criteria (Explicit)**

	<b>Have lessons learned been documented?</b>
	<b>Have required work products been developed?</b>
	<b>Has the AE process been developed?</b>
	<b>Has the AE process been enacted in the SEE?</b>
	<b>Have the AE processes been followed?</b>
	<b>Have the functional requirements for Nav/Com been met?</b>
	<b>Were the requirements developed in the time allocated?</b>
	<b>Were metrics collected?</b>
	<b>Were processes documented?</b>



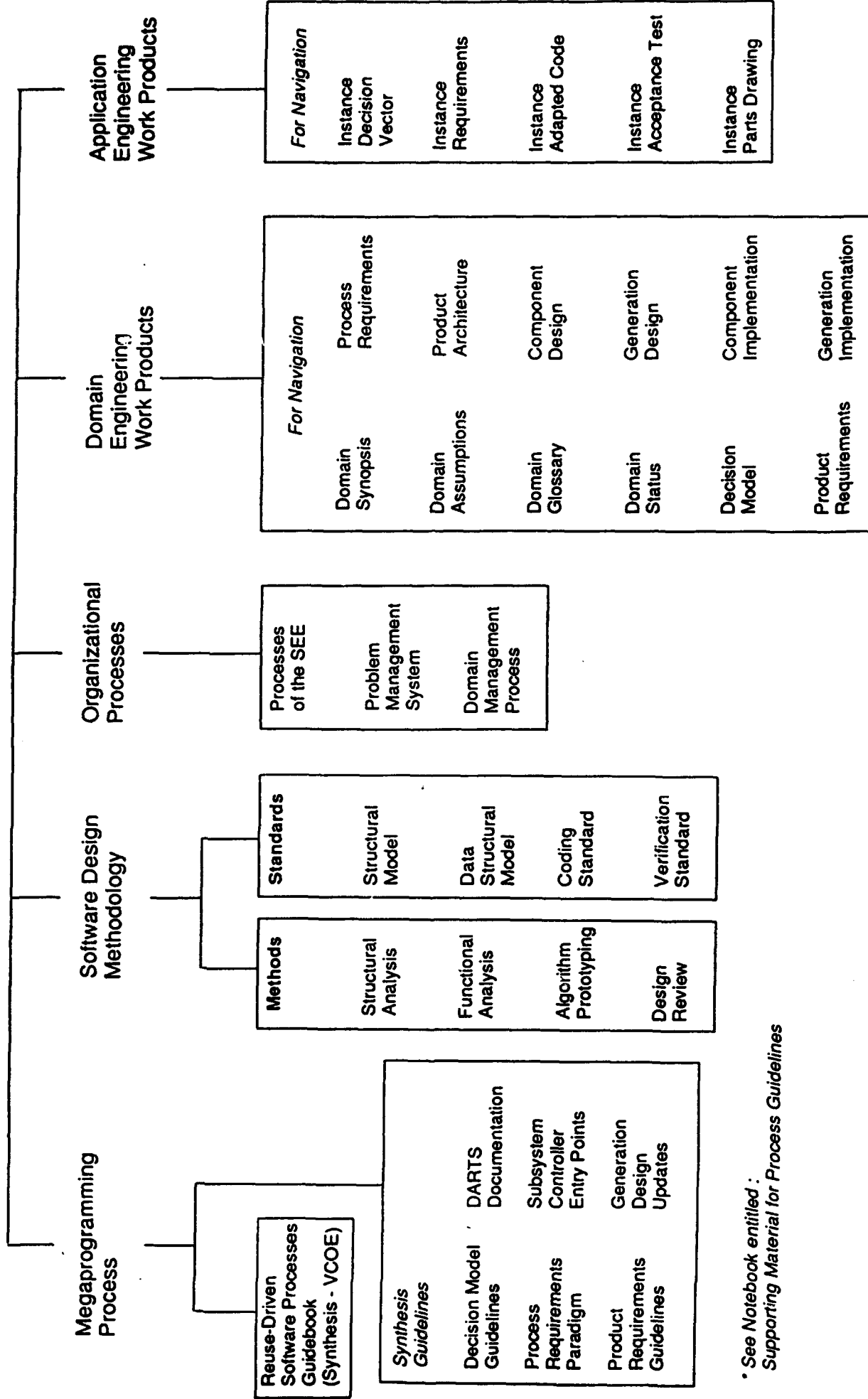
# Success Criteria (Implicit)

Is the team ready for the Performance Phase?

	The team has a complete set of processes
	The team has confidence in the effectiveness of those processes and the capability to perform them
	There is effective automated support for these processes
	The team understands remaining risks and has a plan to control their impact
	The team has a realistic, agreed to, and a visible plan to accomplish the demo
	The team has the critical resources and a credible plan for acquiring additional resources
	Senior leadership is committed to giving mega programming a fair test

# AVTS Pilot Demonstration Project Document Tree

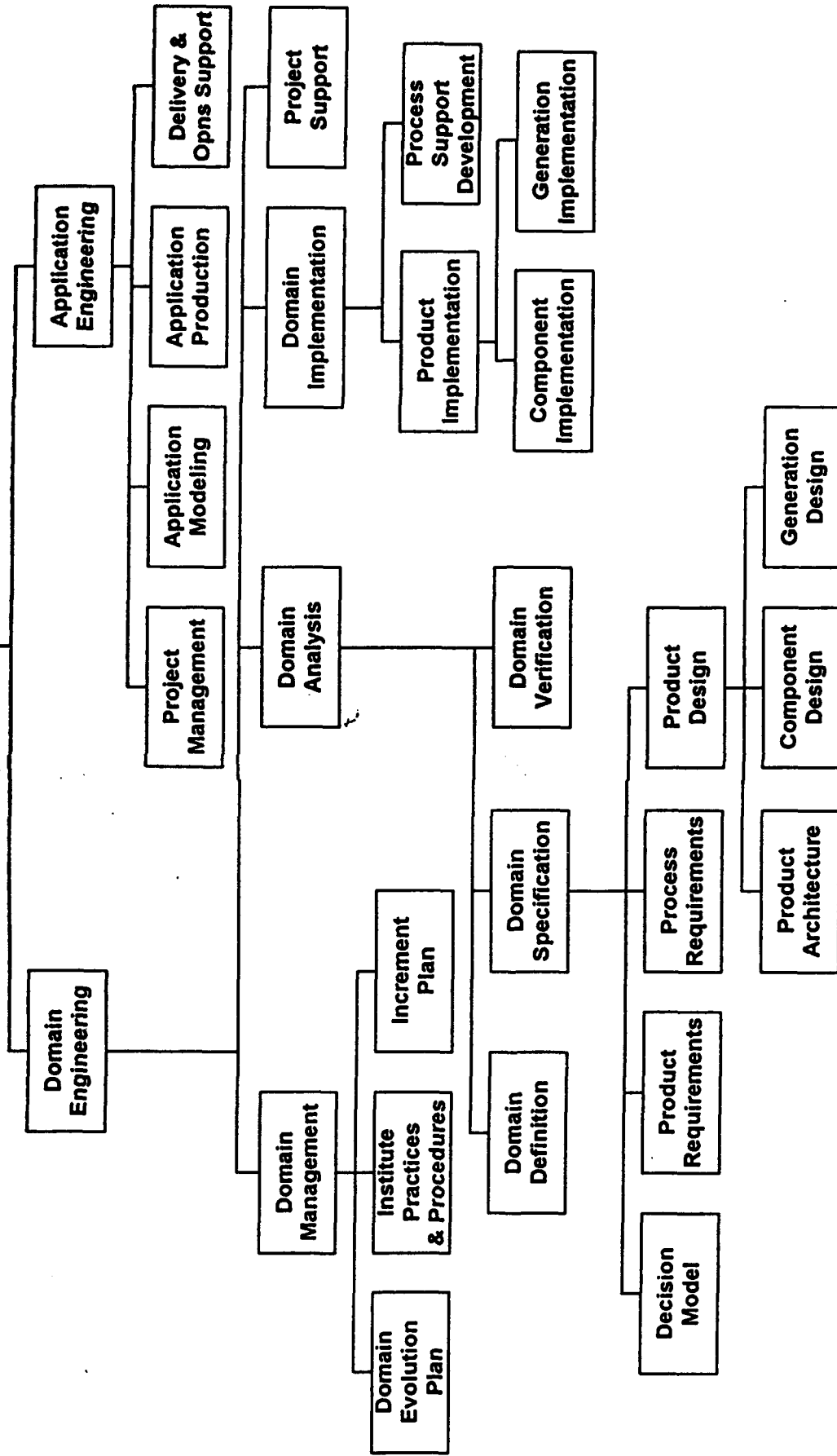
6

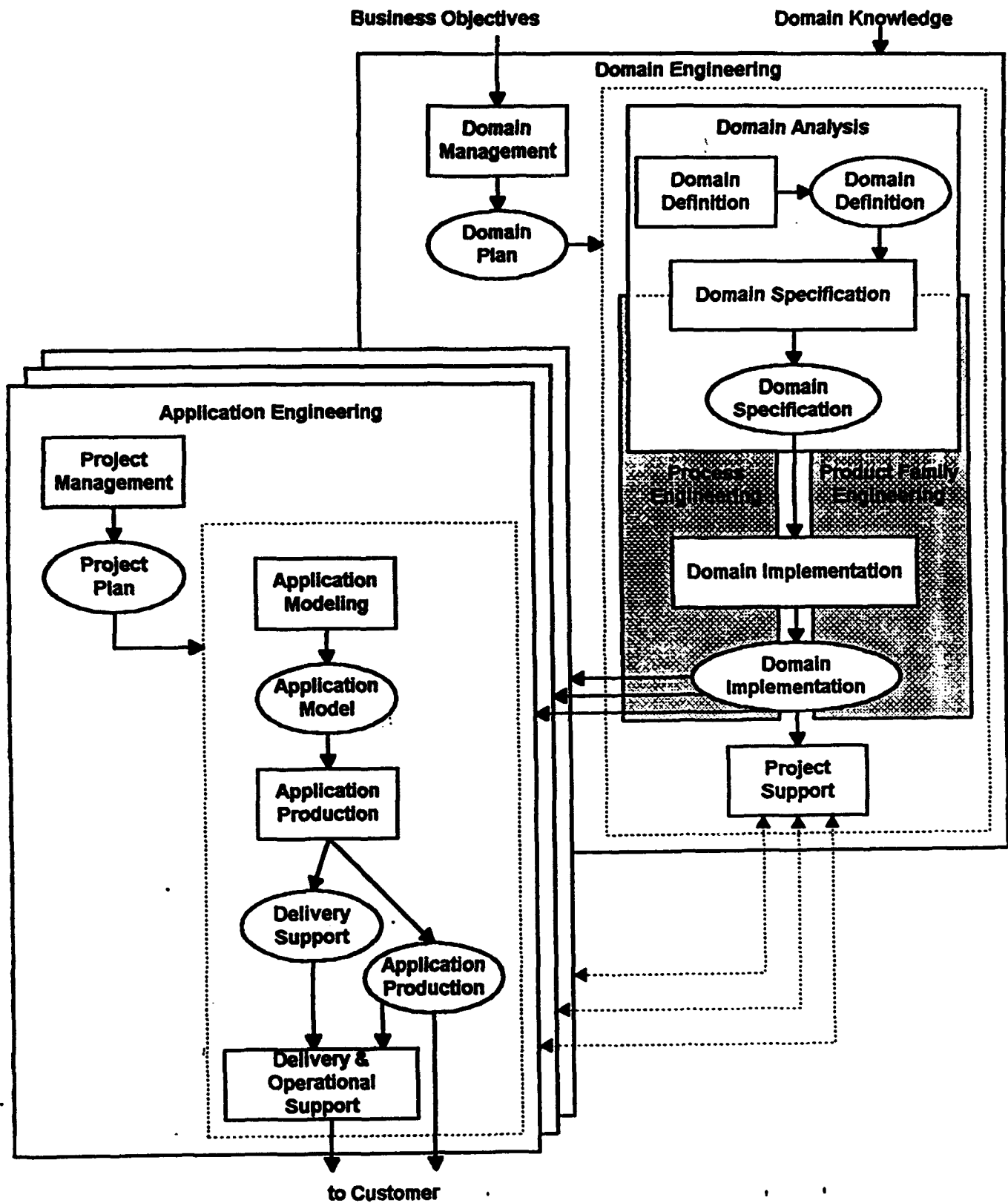


\* See Notebook entitled :  
Supporting Material for Process Guidelines

# Leveraged Synthesis

7





## **Domain Management**

---

**Purpose:** Responsible for managing business area resources to achieve design business objectives

**Product:** Domain Plan

- Domain Evolution Plan - Long Term
- Domain Increment Plan - Near Term
- Practices and Procedures

**Lessons Learned:**

- Formal Domain Plan is essential

## Domain Definition

---

- **Concepts:**
  - Description of the scope of the domain
  - Definition of the terminologies
  - Identification of commonalities and variabilities
  - Determine technical maturity of the domain
  - Determine exclusions within the domain
- **Work products examples**
  - Synopsis ✓ - *skt*
  - Glossary ✓ - *skt*
  - Assumptions
  - Status ✓ - *skt*
- **Success indications**
  - Work products developed
  - Lessons learned documented
    - Define variability early
    - Standardize terminologies (Synthesis/Darts)

## Domain Analysis

---

- **Concepts:**
  - Business Case
  - Domain expertise
  - Process and product development for use by an Application Engineer
- **Examples Work Products:**
  - Domain Definition
  - Domain Specification
- **Success Indicators**
  - All success criteria were met

# Navigation/Communication Domain Synopsis

## NAVIGATION/COMMUNICATION ASSUMPTIONS

### COMMONALITIES .

...

2.5 Self test for TACAN radios have the following commonalities:

- o Each has a set of discrete events that occur over a finite amount of time.
- o There is a way to manually initiate self test.
- o Self test results in TACAN outputs being driven to predefined values.

*Justification:* Based on analysis of self test characteristics using 3 aircraft as examples. (T34, T44, T45).

...

### VARIABILITIES

...

2.1.10 Self Test characteristics vary for each type of radio in the domain.

2.1.10.1 The ways that self test is initiated varies for each type of radio.

2.1.10.2 The duration of self test varies for each type of radio.

2.1.10.3 The characteristics that are exhibited during self test varies for each radio.

2.1.10.4 The way self test is terminated varies for each type of radio.

2.1.10.5 The interaction of TACAN/VOR self test with other systems varies for given cockpit configurations.

2.1.10.6 Self\_Test may or may not exist for TACAN/VOR.

*Justification:* For real aircraft in the domain, detailed self test characteristics will vary.

...

### EXCLUSIONS

# DOMAIN ASSUMPTIONS



---

## Decision Model

---

- **Concepts:**
  - Basis (foundation) for the Application Engineering process
  - Distinguishes leverage reuse through identification variabilities in the form of questions
  - Basis for "tailoring" generics to specifics
- **Examples work products within the Decision Model:**
  - Decision Tables - a set of decisions representing the variability assumptions
  - Decision Group - a set of related decisions
  - Decision Constraints - Corresponds to the number of possible decisions.
- **Success indications:**
  - Work products were produced
  - Functionality was addressed
  - Lessons learned
    - Domain experts are essential
    - Design of Decision Model should be driven by how best to represent variability, not presentation format
    - Decision Groups should be kept small

TACAN Self Test Decision Variables				
AVTS.NAVCOM.RAIDS.TACAN[for each TACAN_Crew_Locations].SLFTST				
Entry Criteria : AVTS.NAVCOM.RAIDS.TACAN [for each TACAN_Crew_Locations] TACAN_SELF_TEST_REQUIRED Must not be FALSE				
Decision Variable	Structural Constraint	DC Ref	VA Ref	Description
Change_In_Channel_Terminate_TACAN_Self_Test	Exactly One (Yes, No)		VA_2.1.10.4	For TACAN Self Test there are several ways to terminate the self test.
Components_Affected_By_Tacancan_Self_Test	Zero or More (VOR, INS, RNAV, ADF)		VA_2.1.10.5	For TACAN there may be other systems that react to a TACAN self test signal
TACAN_Self_Test_Initialized	One or More (Push and Hold, Push and Release, Power On, Other)		VA_2.1.10.1	For initiating self test there are several options. These options are based on the type of control unit under consideration.
TACAN_Button_Pressed_During_Self_Test	Exactly One (Restart, Terminate, Nothing)	DC_2	VA_2.1.10.1	For TACAN self test initiated by a push and type button, there is variability in the effect of pressing the button again during self test.
TACAN_Self_Test_Phases	Integer in Range [0,25]		VA_2.1.10.2	In a TACAN radio, there are a variable number of discrete events that occur during self test. Each phase is defined by a change in the characteristics of self test.

## DECISION MODELS

## **Product Requirements**

---

- **Concepts:**
  - Describes the product family scope & behavior
  - Describes the purpose and objective of the product family
  - Describes the relationship within the product family
  - Describes the limits of the product family
  - Products and Process Requirements can be done concurrently (and was)
- **Example work products**
  - Product Requirements Specification
- **Success indications:**
  - Functional requirements were developed
  - Requirements were developed in the time allocated
  - Work products were developed

# PRODUCT REQUIREMENTS

(B)

## Navigation/Communication Product Requirements

### 1. INTRODUCTION

### 2. REQUIREMENTS.

#### 2.1 Segment Definition.

#### 2.2 Characteristics.

##### 2.2.1 Performance Characteristics.

##### 2.2.1.1 Segment Modes and States.

2.2.1.2 Nav/Comm Segment Functions. The following functions shall be accomplished by the Nav/Comm segment.

##### 2.2.1.2.1 Nav/Comm Segment Support Function.

*if Radio\_Navigation\_Aids then*

*if VHF\_Nav\_System then*

*if VHF\_Nav\_Self\_Test\_Required then*

There shall be a self test function for the VHF Nav system.

*if VOR\_Self\_Test\_Initiated (Push and Hold) then*

The VOR self test shall simulate the characteristics of self test by pressing and holding the self test switch .

*end if*

*end if*

*end if*

*if TACAN\_System then*

*if TACAN\_Self\_Test\_Required then*

There shall be a self test function for the TACAN system.

*if TACAN\_Self\_Test\_Initiated (Push and Hold) then*

The TACAN self test shall be initiated by a press and hold button.

*end if*

*end if*

*end if*

*end if*

## **Process Requirements**

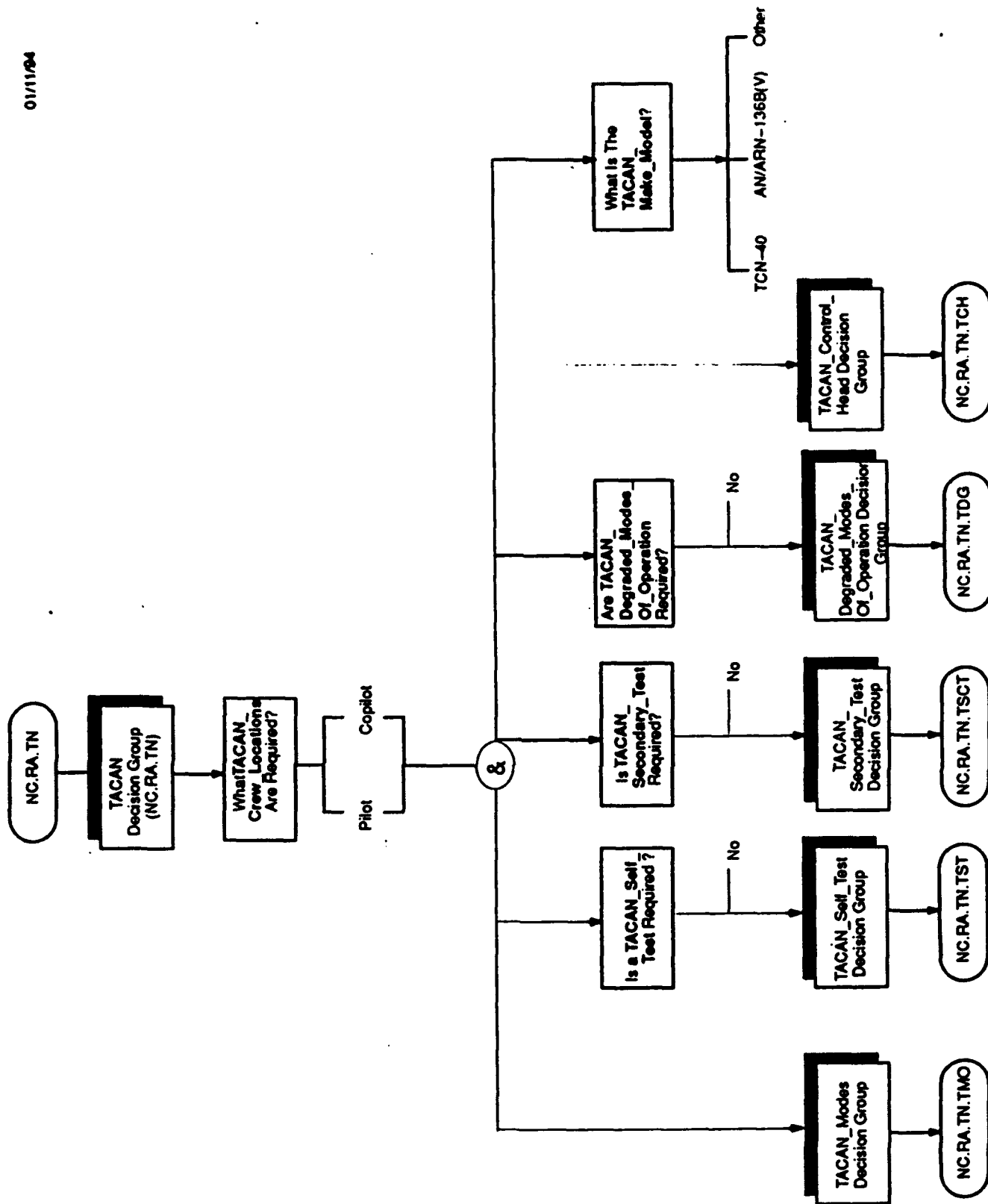
---

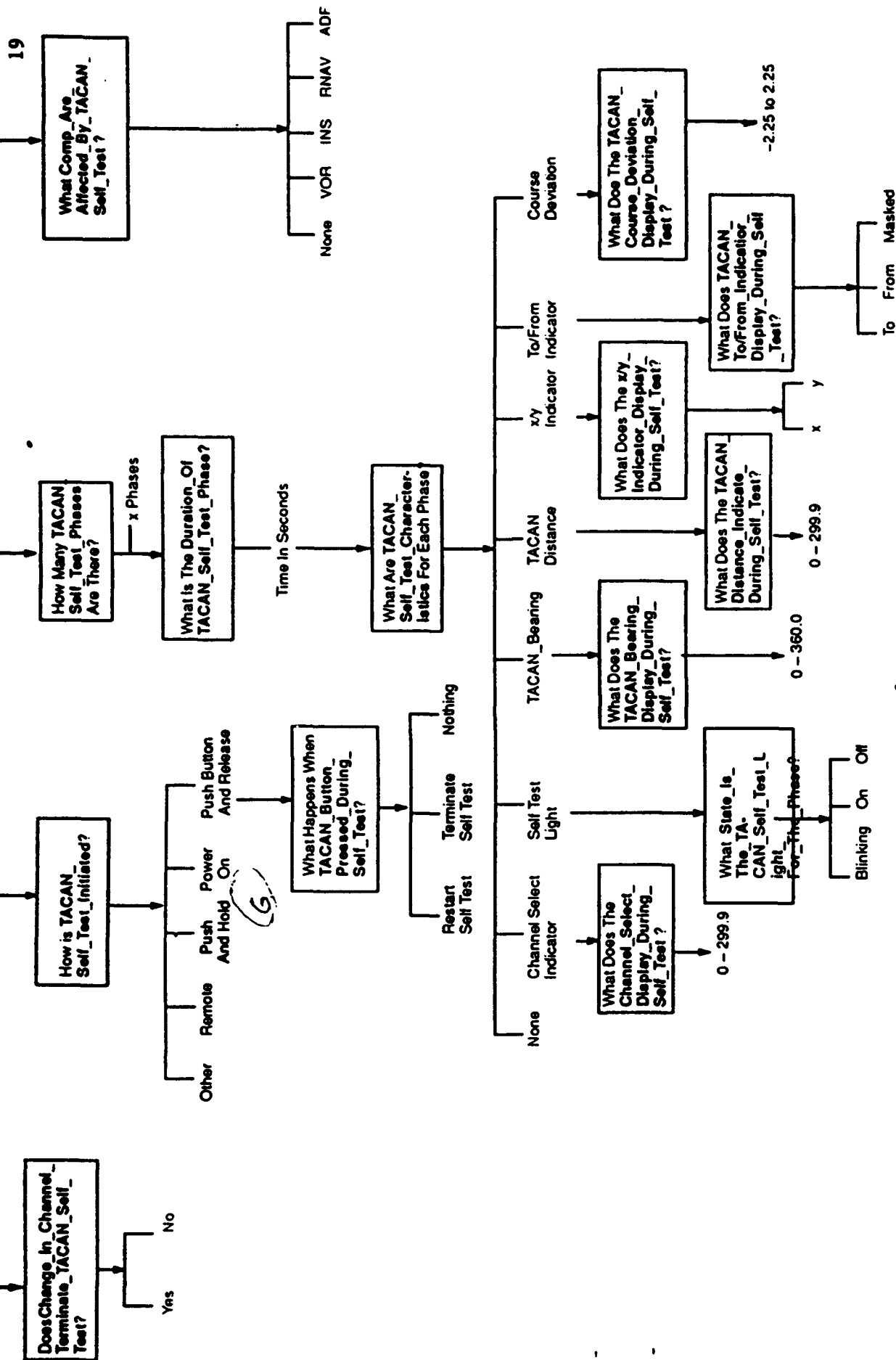
- **Concepts:**
  - Describes the Application Engineering process
  - Describes the procedures for identifying reuse
- **Examples work product:**
  - Process Specification
  - Application Modeling Notation Specification
- **Success indications:**
  - Work products were developed
  - Application Engineering process was developed
  - Application Engineering process was documented

# TACAN DECISION GROUP

18

01/11/94





## **Product Design**

---

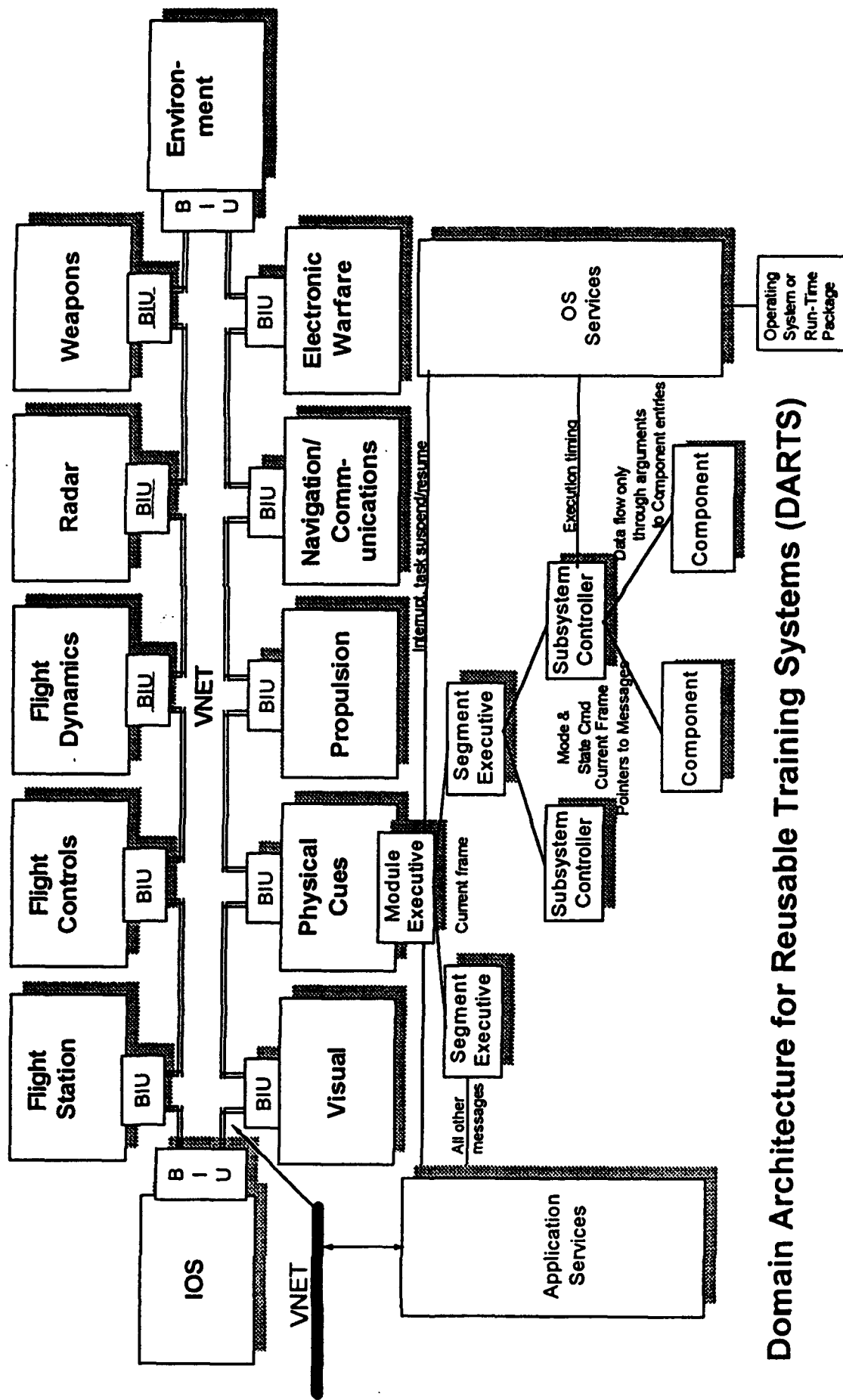
- **Concepts:**
  - Specifies the design of the product family
  - Development of design methodologies
- **Example work products:**
  - Product Architecture
  - Component Design
  - Generation Design
- **Success Indications**
  - Work products have been developed
  - Lessons learned have been documented
    - Design methodologies are required as entrance criteria
    - Design should allow for feedback loop between component design and architecture



## Architecture

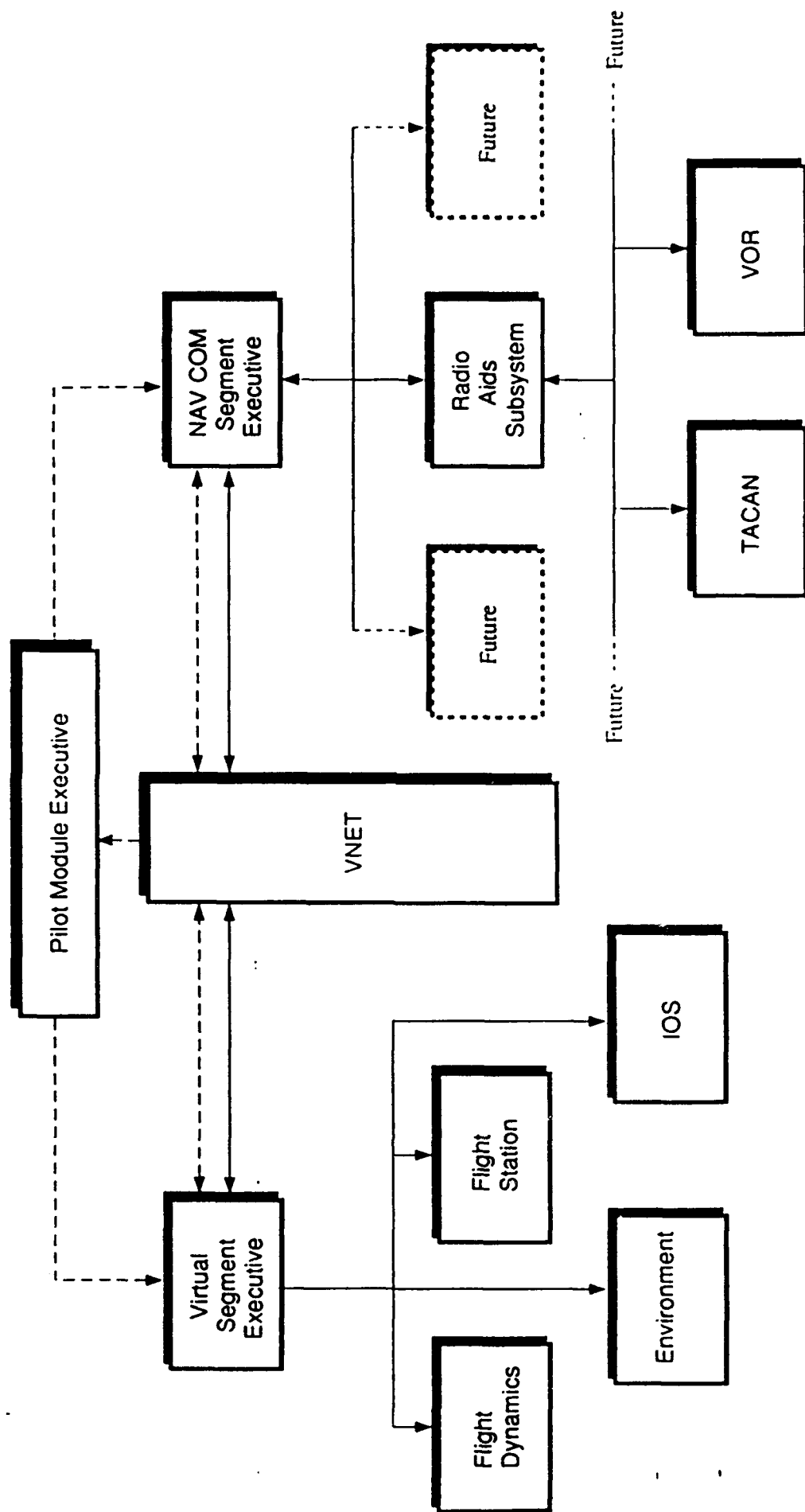
---

- **Concepts:**
  - Adopted a mature architecture for the process (DARTS)
  - Supported by design methods (flow, dependency & object) developed by the team
  - AVTS (DARTS message handler) central to overall structure
- **Example work products:**
  - Product Architecture
- **Success indication:**
  - Work products have been developed
  - Lessons learned have been documented
    - Product Architecture must provide necessary levels of encapsulation
    - Product Architecture can be fine grained
    - Architecture does not necessarily match the Decision Model



Domain Architecture for Reusable Training Systems (DARTS)

# Slice Simulator Architecture



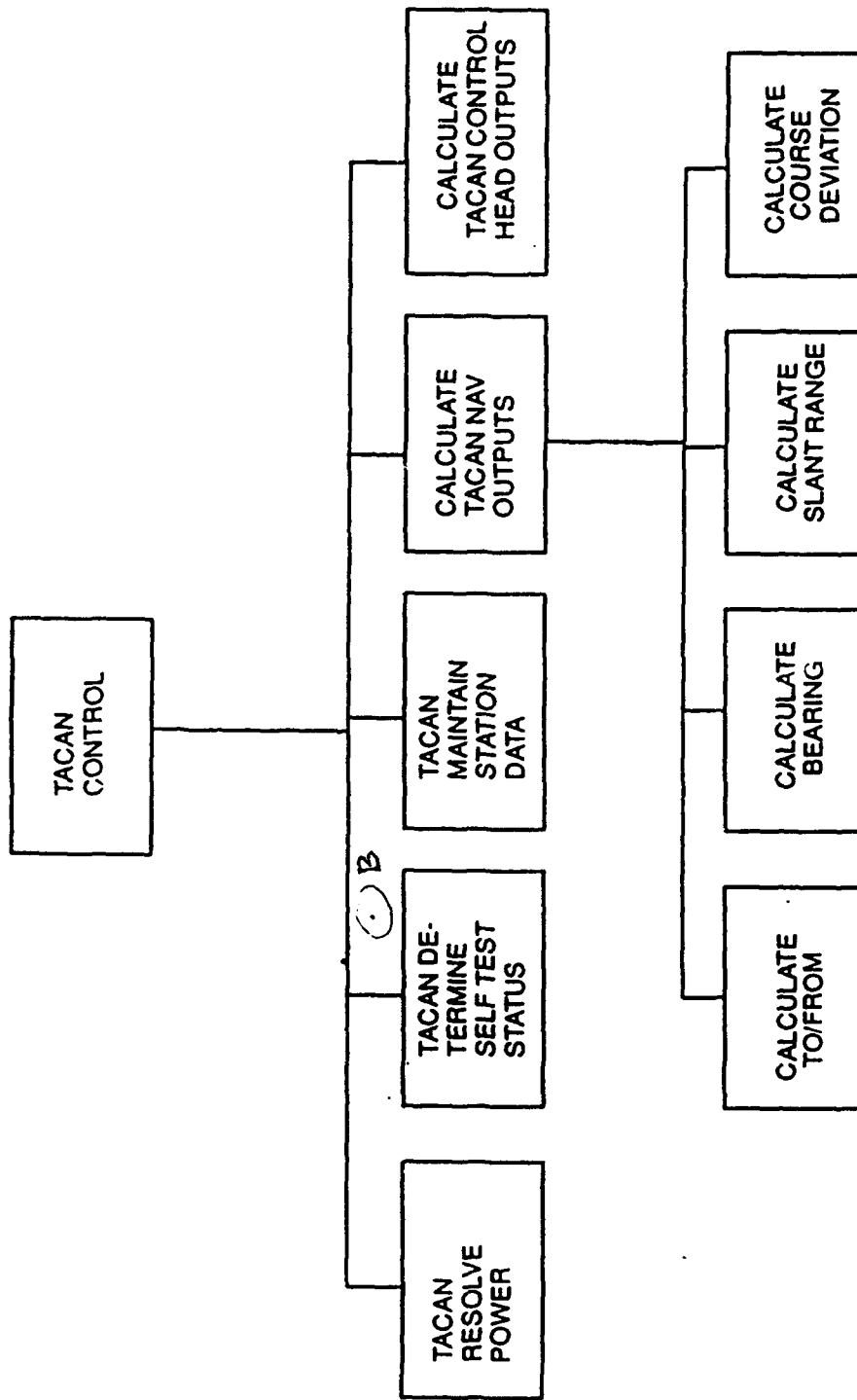


FIGURE 2.1.3 TACAN ARCHITECTURE

## **Component Design**

---

- **Concept:**
  - Component design is similar to traditional design
  - Component design defines the interfaces
- **Example work products**
  - Adaptation Specification
  - Interface Specification
- **Success indications:**
  - Work products were developed
  - Lessons Learned were documented
  - Component Design effects product architecture and vice versa

```

    self_test_active = true          — set self test to active
  endif
endif
<else if P_TAC_ST_PUSH_AND_HOLD> — if it's a press and hold push button.
  if self_test_activate = true      — Self test button pressed then
    self_Test_Activate = true       — set self test to active>
  endif
<else>
  null; — none or other manual self test initiation
<endif>

— process self test
if self test active
  increment self_test_timer;

  — check for premature termination
  <if P_TAC_TERM_WITH_CHAN then> — if TACAN self test terminated by change in
    — channel.
    if change_in_channel then
      terminate_self_test
    endif
  <endif>

  <if other termination condition then>
    Terminate_self_test
  <endif>

  — now determine the phase
  phase_found = false              — initialize a flag for search loop.
  loop for each phase              — for each phase of time that's defined in self test.
    if timer < phase end time then — see if the timer falls into that time slot.
      phase = loop index;          — set the phase found flag
      phase_found = true           — set the phase found flag
    exit loop;
  endloop
  if phase_found = false           — if all phases have run to completion.
    terminate_self_test;
  endif

endif; — end process self test logic

procedure Terminate_Self_Test;

```

# COMPONENT DESIGN

## **Generation Design**

---

- **Concepts:**
  - Specifies the mapping that will produce Application Engineering work products
  - Generated Design is the adaptation of the Component Design
- **Example mappings**
  - Architecture Mapping - Decision Model-to-Architecture
  - Component Mapping - How each component of a work product is to be produce
  - Decision Mapping - Relations between Decision Model and instantiation parameters for the Component Design
- **Success indications**
  - Work products were developed
  - Lessons learned were documented
  - Process support is required

P_INITIATE_PUSH_AND_HOLD	if (Push_and_Hold ∈ %loc%_%inst%_%tst%_Test_Initiated ) then TRUE else FALSE
P_INITIATE_REMOTE	if (Remote ∈ %loc%_%inst%_%tst%_Test_Initiated ) then TRUE else FALSE
P_AFFECT_OF_REPRESS_RESTART	if (%loc%_%inst%_Button_Pressed_During_%tst%_Test = Restart) then TRUE else FALSE
P_AFFECT_OF_REPRESS_TERMINATE	if (%loc%_%inst%_Button_Pressed_During_%tst%_Test = Terminate) then TRUE else FALSE
P_AFFECT_OF_REPRESS_NO_EFFECT	if (%loc%_%inst%_Button_Pressed_During_%tst%_Test = Nothing) then TRUE else FALSE
P_ST_FAIL	if (Self_Test_Fail ∈ TACAN_Malfunctions) then TRUE else FALSE

# DECISION MAPPING

2-13 T



## Generation Design

**Architecture Map** – provides the mapping between the architecture nodes and the instantiation parameters. This defines what architectural nodes are required given an application model (instance of the domain).

**NAV/COMM Segment Exec**

If Navigation\_Communication\_System = Yes

**Radio Navigation Aids**

If Radio\_Navigation\_Aids = Yes

**Pilot TACAN Control**

If TACAN\_Systems = Yes and {Pilot is an element of TACAN\_Crew\_Locations}

**Pilot TACAN Resolve Power**

If TACAN\_Systems = Yes and {Pilot is an element of TACAN\_Crew\_Locations}

**Pilot TACAN Determine Self Test Status**

If Pilot\_TACAN\_Self\_Test\_Required = Yes

**Pilot TACAN Maintain Station Data**

If TACAN\_Systems = Yes and {Pilot is an element of TACAN\_Crew\_Locations}

**Calculate Pilot TACAN Nav Outputs**

If TACAN\_Systems = Yes and {Pilot is an element of TACAN\_Crew\_Locations}

**Calculate Pilot TACAN Control Head Outputs**

If TACAN\_Systems = Yes and {Pilot is an element of TACAN\_Crew\_Locations}

**Pilot VOR Control**

If VHF\_Nav\_System = Yes and {Pilot is an element of VOR\_Crew\_Locations}

**Pilot VOR Resolve Power**

If VHF\_Nav\_System = Yes and {Pilot is an element of VOR\_Crew\_Locations}

**Pilot VOR Determine Self Test Status**

If Pilot\_VHF\_Nav\_Self\_Test\_Required = Yes

**Pilot VOR Maintain Station Data**

If VHF\_Nav\_System = Yes and {Pilot is an element of VOR\_Crew\_Locations}

**Calculate Pilot VOR Nav Outputs**

If VHF\_Nav\_System = Yes and {Pilot is an element of VOR\_Crew\_Locations}

**Calculate Pilot VOR Control Head Outputs**

If VHF\_Nav\_System = Yes and {Pilot is an element of VOR\_Crew\_Locations}

# ARCHITECTURE MAPPING

# COMPONENT MAPPING

5-1235

**Component Map** – provides the mapping between the architectural nodes and the adaptable components that fill the node.

**NAV/COMM Segment Exec**

Use Component: Package, Navigation\_Communication\_Executive

**Radio Navigation Aids**

Use Component: Package, Radio\_Nav\_Aids\_Subsystem\_Controller

**Pilot TACAN Control**

Use Component: Package, TACAN

**Pilot TACAN Resolve Power**

B  Use Component: Procedure, TACAN\_Resolve\_Power

**Pilot TACAN Determine Self Test Status**

Use Component: Package, Self\_Test

**Pilot TACAN Maintain Station Data**

Use Component: Procedure, Maintain\_Station\_Data

**Calculate Pilot TACAN Nav Outputs**

Use Component: Procedure, TACAN\_Nav\_Outputs

**Calculate Pilot TACAN Control Head Outputs**

Use Component: Procedure, TACAN\_Control\_Head\_Outputs

**Pilot VOR Control**

Use Component: Package, VOR

**Pilot VOR Resolve Power**

Use Component: Procedure, VOR\_Resolve\_Power

**Pilot VOR Determine Self Test Status**

Use Component: Package, Self\_Test

**Pilot VOR Maintain Station Data**

Use Component: Procedure, Maintain\_Station\_Data

**Calculate Pilot VOR Nav Outputs**

Use Component: Procedure, VOR\_Nav\_Outputs

**Calculate Pilot VOR Control Head Outputs**

Use Component: Procedure, VOR\_Control\_Head\_Outputs

## **Domain Implementation**

---

- **Key activities:**
  - Product Implementation
    - Component Implementation
    - Generation Implementation
  - Process Support Development
- **Success indication:**
  - Work products have been developed
  - Application Engineering process was documented
  - Application Engineering process was developed
  - Application Engineering process exacted in SEE
  - Lessons learned were documented
    - ADA skills are required

## **Component and Generation Implementation**

---

- **Concepts:**
  - Implement components with variables as instantiation parameters
  - Create automated processes to:
    - Capture variations as questions
    - Retrieve and adapt components
- **Examples:**
  - Adaptable components
  - Generation procedures
- **Success indications**
  - Work products have been developed
  - Application Engineering process has been developed
  - Lessons learned have been documented
    - Generation Implementation should be performed by a process engineer

33 (deffunction make-TACANSelfTest (?spawninst ?value)

```
(if (eq ?TACAN_C_answer_value Yes) then
  (bind ?value Copilot)
  (bind ?nvalues (+ ?nvalues 1) )
)
(if (eq ?TACAN_P_answer_value Yes) then
  (bind ?value Pilot)
  (bind ?nvalues (+ ?nvalues 1) )
)
.
```

```
(bind ?newsym (sym-cat ?value "TACSlfTestInit"))
(bind ?newdesc (str-cat ?value "TACSlfTestInit"))
```

```
(bind ?newnam (symbol-to-instance-name ?newsym))
(bind ?qnewsym (sym-cat ?newsym "Question"))
(bind ?qnewnam (symbol-to-instance-name ?qnewsym))
```

```
(make-instance ?newnam of MULTI_DECISION
  (Design_Group_ID TACANSelfTest)
  (Question_Name ?qnewsym)
  (One_Line_Desc ?newdesc)
  (InstanceInstantiatedBy ?spawninst)
  (FurtherQFunction make-buttonpress)
  (Multi_Entry nil))
```

```
(make-instance ?qnewnam of MULTIPLE_CHOICE_QUESTION
  (Decision_Name ?newsym)
  (Question (format nil "%s%s%n%s" "How is " ?value
    "TACAN self test Initiated ?"))
  (One_Line_Question ?newdesc)
  (Text_Choices Push_And_Hold Power_On Push_And_Release Other)
  (Choices Push_And_Hold Power_On Push_And_Release Other)
  (Lines_Selected 0)
  (Decision_Help_Text (format nil "%s%n%s%n%s"
    "For initiating self test there are"
    "several options. These options are based on "
    "the type of control unit under consideration"))
  .
  .
  .
```

# KAMEL MAKE FUNCTION

# DECIDE SCREEN

34

2-15 T

File Actions

Help

How Is Pilot

TACAN self test Initiated?

- ☒ Push\_And\_Hold ☐ Push\_And\_Release  
☐ Power\_On ☐ Other

Malfunctions : Yes  
BackdoorInterface : No  
Diagnostics/Test : No  
Segment--Occulting : Environment  
Segment--RadarDB/GA : None  
Segment--VisualDB/GA : None  
Segment--SpatialRels : Environment  
Scoring : No  
Autotest Capab. : No  
Reposition Capab. : Yes  
Motion Fidelity : 6\_DOF  
Engine Type : Turbine  
Air Vehicle Class : Airplane  
Training Sys Name : Flight Instrument

Tacan Self Test Decision Group : Enac!  
PilotTACSIlfresterm : No

Accept

Decision Help

Re--Decide

2-15

```

(bind ?tests_a (create$ TAC TAC VOR VOR))
(bind ?tests_b (create$ Slf Sec Slf Sec))
(bind ?tests_c (create$ SELF SECONDARY SELF SECONDARY))
35 (bind ?tests_d (create$ TACAN TACAN VOR VOR))

```

```

(while (<= ?testtype 4 )

```

```

    (bind ?tal (nth ?testtype ?tests_a))
    (bind ?tbl (nth ?testtype ?tests_b))

```

```

    .
    .
    .

```

```

;--- Check for initiate at: power on, push & release, push &hold

```

```

(bind ?testinit (sym-cat "Pilot" ?tal ?tbl "TestInit"))
(bind ?testinitinst (symbol-to-instance-name ?testinit ))
(bind ?testinitans (send ?testinitinst get-Multi_Entry ))

```

```

(if ( member Power_On ?testinitans ) then
    (bind ?P_INITIATE_AT_POWER_ON TRUE )
else
    (bind ?P_INITIATE_AT_POWER_ON FALSE )
)

```

```

(if ( member Push_And_Release ?testinitans ) then
    (bind ?P_INITIATE_PUSH_AND_RELEASE TRUE )
else
    (bind ?P_INITIATE_PUSH_AND_RELEASE FALSE )
)

```

```

(if ( member Push_And_Hold ?testinitans ) then
    (bind ?P_INITIATE_PUSH_AND_HOLD TRUE )
else
    (bind ?P_INITIATE_PUSH_AND_HOLD FALSE )
)

```

```

    .
    .
    .

```

```

(format ?where "%s%s%n" " ADAPT $P_INITIATE_PUSH_AND_HOLD$ "
    ?P_INITIATE_PUSH_AND_HOLD )

```

```


    .
    .
    .

```

# KAMEL RETRIEVE FUNCTION

# 1 APPLICATION MODEL/DECISION-VECTOR

## 1.1 Decision Group : TACAN\_SELF\_TEST

- PilotTACSlfTestTerm = No 
- PilotTACSlfTestInit = Push\_And\_Hold
- PilotTACSlfTestPhases = 1
- PilotTACCompAffect = VOR
- PilotTACSlfPhase1Ch = Channel\_Select\_Indicator X/Y\_Indicator
- PilotTACSlfPhase1ChChSlIn = 188.0
- PilotTACSlfPhase1ChXYInd = X



## **Process Support Development**

---

- **Concepts:**
  - Translates and "load" Domain Engineering work products in the SEE
  - Procedural standards
- **Example Work Products**
  - Application Engineering Process Standards
  - Application Engineering User's Guide
  - Application Engineering Environment
  - Application Engineering Environment Support Manual
  - Application Engineering Training Course
- **Success indications**
  - Application Engineering process has been enacted in the SEE
  - Lessons learned have been documented
  - SEE enactment process must be mature to be followed properly

Determined\_Active := false; (B)

-- Look at TACAN power. Dont execute if power is off.  
 if Power = On then

--\$-- Include if auto self test at power up.  
 --\$if SP\_INITIATE\_AT\_POWER\_ONS THEN  
 if New\_Power = base\_types.true then  
 Determined\_Active := true;  
 end if;  
 --Send IF

--\$-- Include if type is push and release  
 --\$if SP\_INITIATE\_PUSH\_AND\_RELEASES THEN  
 if Test\_Activate = On then -- if "button" is currently pressed.  
 Determined\_Active := true;

--\$-- Include if button restarts the test if test is already running.  
 --\$if SP\_AFFECT\_OF\_REPRESS\_RESTARTS THEN  
 if Last\_Pass\_Active then -- if self test is active  
 Timer := 0.0; -- reset timer.  
 end if;  
 --Send IF;

--\$if SP\_AFFECT\_OF\_REPRESS\_TERMINATES THEN  
 if Last\_Pass\_Active then -- if self test is active  
 Determined\_Active := false; -- Deactivate Self Test.  
 end if;  
 --Send IF;

else  
 -- if active from last pass let the timer terminate the test.  
 if Last\_Pass\_Active then -- if active from last pass let  
 Determined\_Active := true;  
 end if;  
 end if;  
 --Send IF

--\$-- Include if type button is push and hold.  
 --\$if SP\_INITIATE\_PUSH\_AND\_HOLDS THEN  
 if Test\_Activate = On then (G)  
 Determined\_Active := true;  
 else  
 Determined\_Active := false;  
 end if;  
 --Send IF

--\$-- Include if change in channel interrupts self test  
 --\$if SP\_TERM\_WITH\_CHANS THEN  
 if Change\_In\_Channel then  
 Determined\_Active := false;  
 end if;  
 --Send IF

if Determined\_Active then

-- If first pass of a self test then initialize timer.  
 if not Last\_Pass\_Active then  
 Timer := 0.0;  
 end if;

-- Increment self test timer.  
 Timer := Timer + Iteration\_Duration;

-- Search phase end time array to determine what phase self test is in  
 for I in 1..Num\_Phases loop


# ADAPTABLE CODE

# ROAMS RETRIEVE FILE

39

-- SELF\_TEST retrieve file

begin

ADAPT \$P_PACKAGE_TEST_NAMES	DETERMINE_TACAN_SELF_TEST
ADAPT \$P_PROCEDURE_TEST_NAMES	PROCESS_TACAN_SELF_TEST
ADAPT \$P_TERM_WITH_CHAN\$	FALSE
ADAPT \$P_EXTERNAL_OUTPUT\$	TRUE
ADAPT \$P_NUM_PHASES\$	"1"
ADAPT \$P_END_TIME_OF_PHASES\$	"0.0"
ADAPT \$P_INITIATE_AT_POWER_ON\$ 	FALSE
ADAPT \$P_INITIATE_PUSH_AND_RELEASE\$	FALSE
ADAPT \$P_INITIATE_PUSH_AND_HOLD\$	TRUE
ADAPT \$P_AFFECT_OF_REPRESS_RESTART\$	FALSE
ADAPT \$P_AFFECT_OF_REPRESS_TERMINATES	FALSE
ADAPT \$P_AFFECT_OF_REPRESS_NO_EFFECT\$	FALSE
ADAPT \$P_ST_FAIL\$	FALSE

COPY type SELF\_TEST\_BODY\_A. TACAN\_SELF\_Test.adb

COPY type SELF\_TEST\_SPEC\_A. TACAN\_SELF\_Test.ads

end

```

-- Declare variables required within package.
-- Number of phases
Num_Phases : Base_Types.Signed_Integer_16 := 1;
Phase : Base_Types.Signed_Integer_16 := 0;
Iteration_Duration : float_32 := 0.1;

Timer : float_32 := 0.0;
Last_Pass_Active : boolean := false;
Test_Active : Base_Types.Discrete_State;

-- Define and initialize array that contains end times for each self test phase.
-- These are elapsed time from self test initiation that each phase will
-- terminate.

type float_array is array (Base_Types.Signed_Integer_16 range <>) of float_32;
Phase_End_Time : float_array (1..1)
:= (
  1 =>
    0.0);

procedure PROCESS_TACAN_SECONDARY_TEST(
  -- Declare variables for this procedure
  Test_Activate : in Base_Types.Discrete_State;
  Power : in Base_Types.Sim_Boolean;
  New_Power : in Sim_Boolean;

  External_Test_Out : out Base_Types.Discrete_State;

  Test_Phase : out Base_Types.Signed_Integer_16) is
-- Declare and initialize local variables.
Phase_Found : Boolean := false;      -- Phase Found flag.
Determined_Active : Boolean := false;

begin
  -- Initialize some variables.
  Phase_Found := false;
  Determined_Active := false;

  -- Look at TACAN power. Dont execute if power is off.
  if Power = True then

    if Test_Activate = On then
      Determined_Active := true;
    else
      Determined_Active := false;
    end if;

    if Determined_Active then
      -- If first pass of a self test then initialize timer.
      if not Last_Pass_Active then
        Timer := 0.0;
      end if;

      -- Increment self test timer.

```

ADAPTED CODE

2-18T

# **Rockwell International**

## **Command and Control Systems Division**

# **Synthesis Pilot Project**

**Presented by**  
**Jerri Turner-Harris**

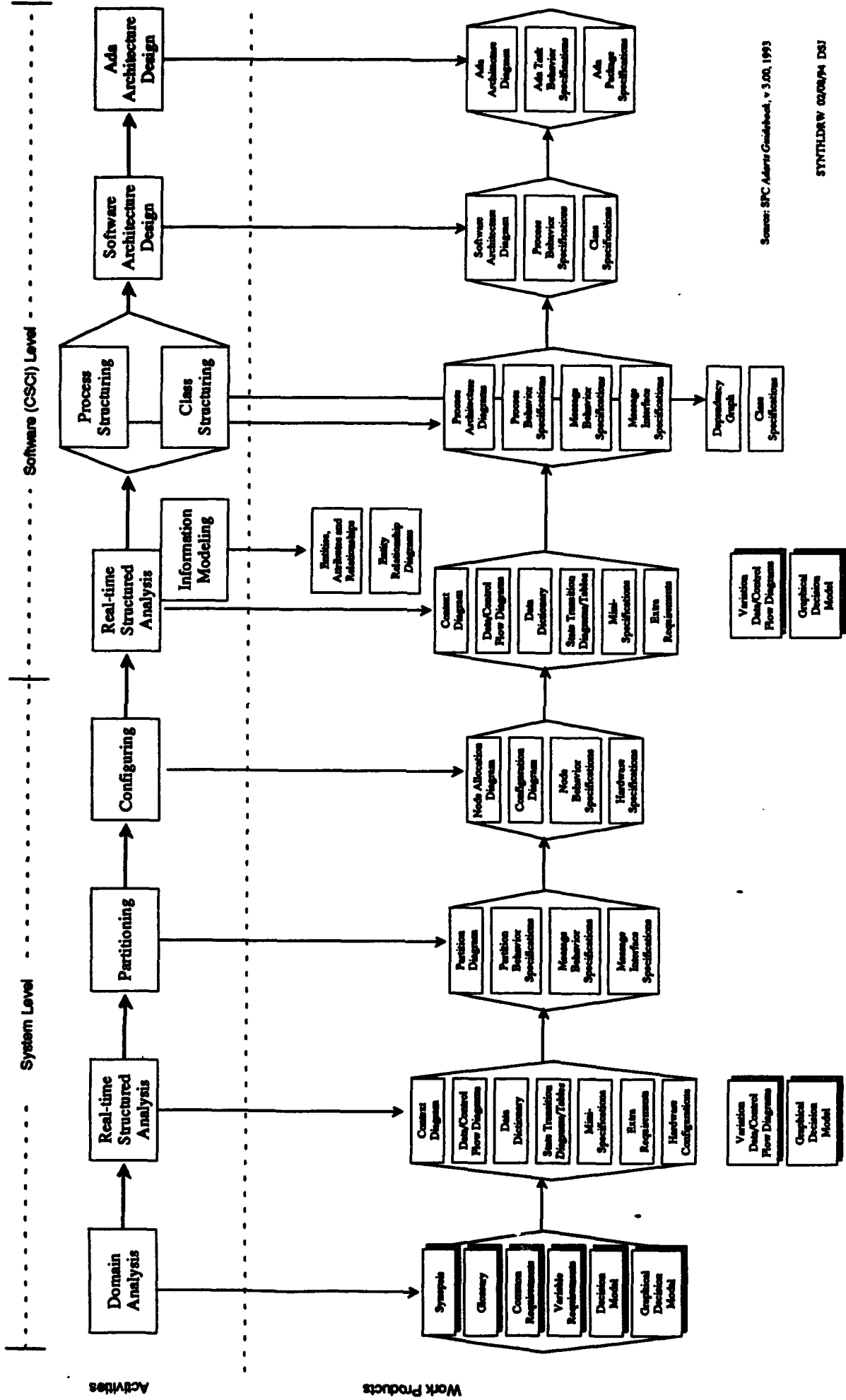
## **DOMAINS / SUBDOMAINS**

- **COMMUNICATIONS MANAGEMENT AND CONTROL  
DOMAIN**
- **MIL-STD-1553B INTERPROCESSOR COMMUNICATION  
SUBDOMAIN**
- **MESSAGE HANDLING SYSTEM DOMAIN**

## **LESSONS LEARNED**

- **ADOPT DISCIPLINED METHODS**  
**REAL-TIME STRUCTURED ANALYSIS**  
**ADARTS**  
**CODING AND DOCUMENTATION STANDARDS**  
**METRICS**
- **COMMUNICATE DOMAIN EXPERIENCE**  
**PRODUCT REVIEWS**
- **USE AVAILABLE AUTOMATED TOOLS**  
**OPENSELECT**  
**TEAMWORK**  
**WORDPERFECT MACRO AND MERGE**  
**TRF**

# The Role of ADARTS in Domain Analysis





## Synopsis Segment

Subdomain implementations provide error free application to application communications by MIL-STD-1553B bus. Subdomain implementations may provide 1553B bus control, 1553B remote terminal and 1553B bus monitoring functions.

### Commonality

Every MHS will periodically delete expired messages from storage.

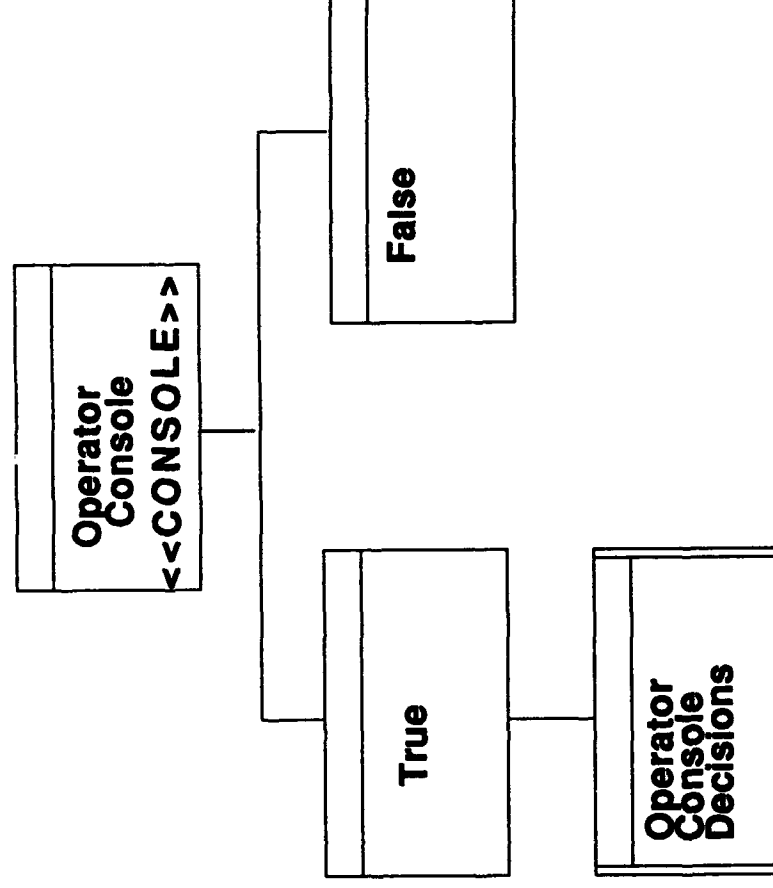
### Variability

- o How long a message can be stored.
- o How often will storage be checked for expired messages.
- o What time will storage be checked.

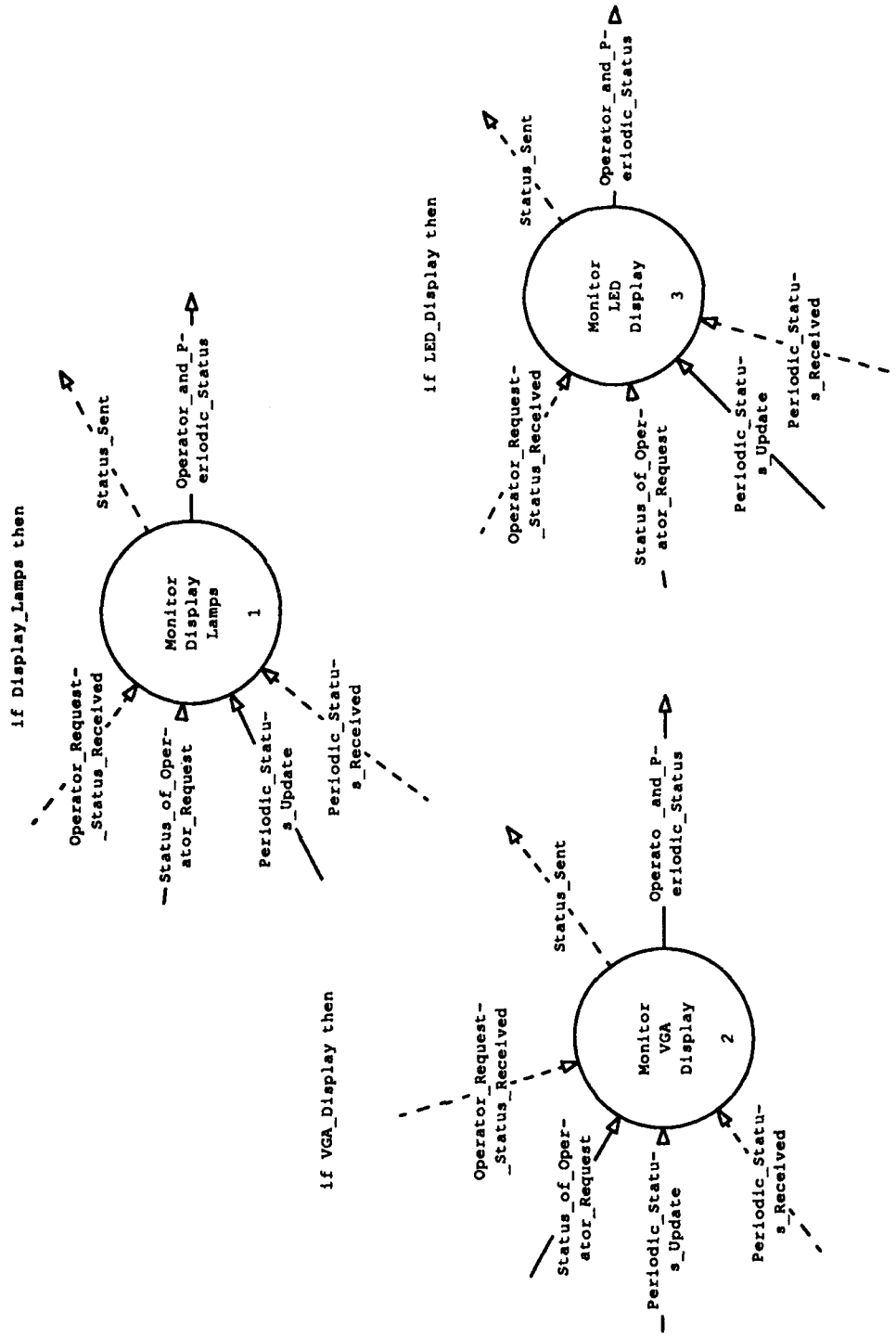
### Decision Model

Description	Decision	Mnemonic
How long a message can be stored.	selection of ( 7 days, 30 days, 60 days, 90 days )	[STORAGE_LENGTH]
How often will storage be checked for expired messages.	selection of ( once a day, once a week, once a month )	[STORAGE_CHECK]
What time will storage be checked.	[(1:00..12:00)] selection of ( am, pm )	[STORAGE_TIME] [STORAGE_AM_PM]

# Graphical Decision Model Segment



## Output Display

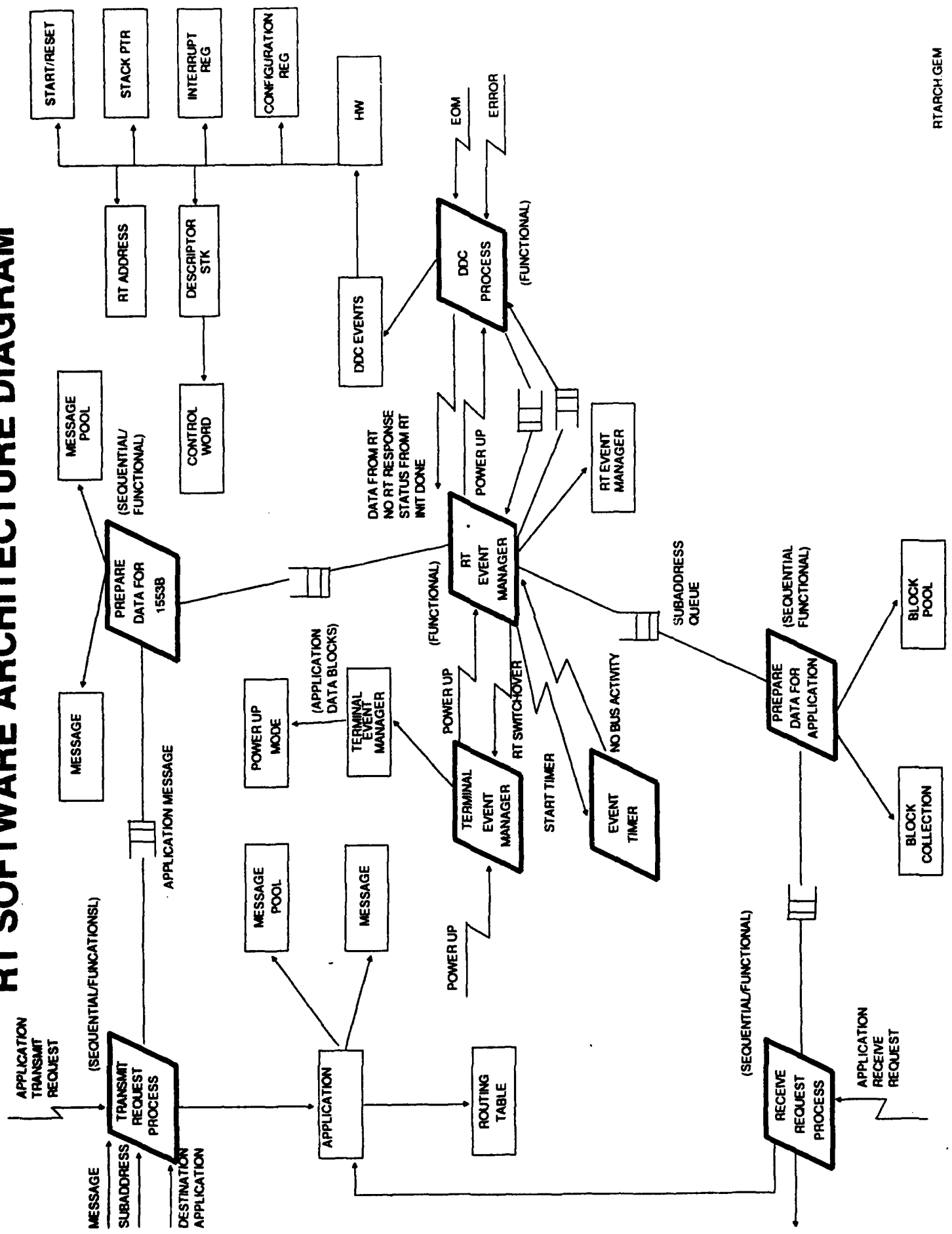
Variation Diagram  
(non-deliverable)

display12.dat

r

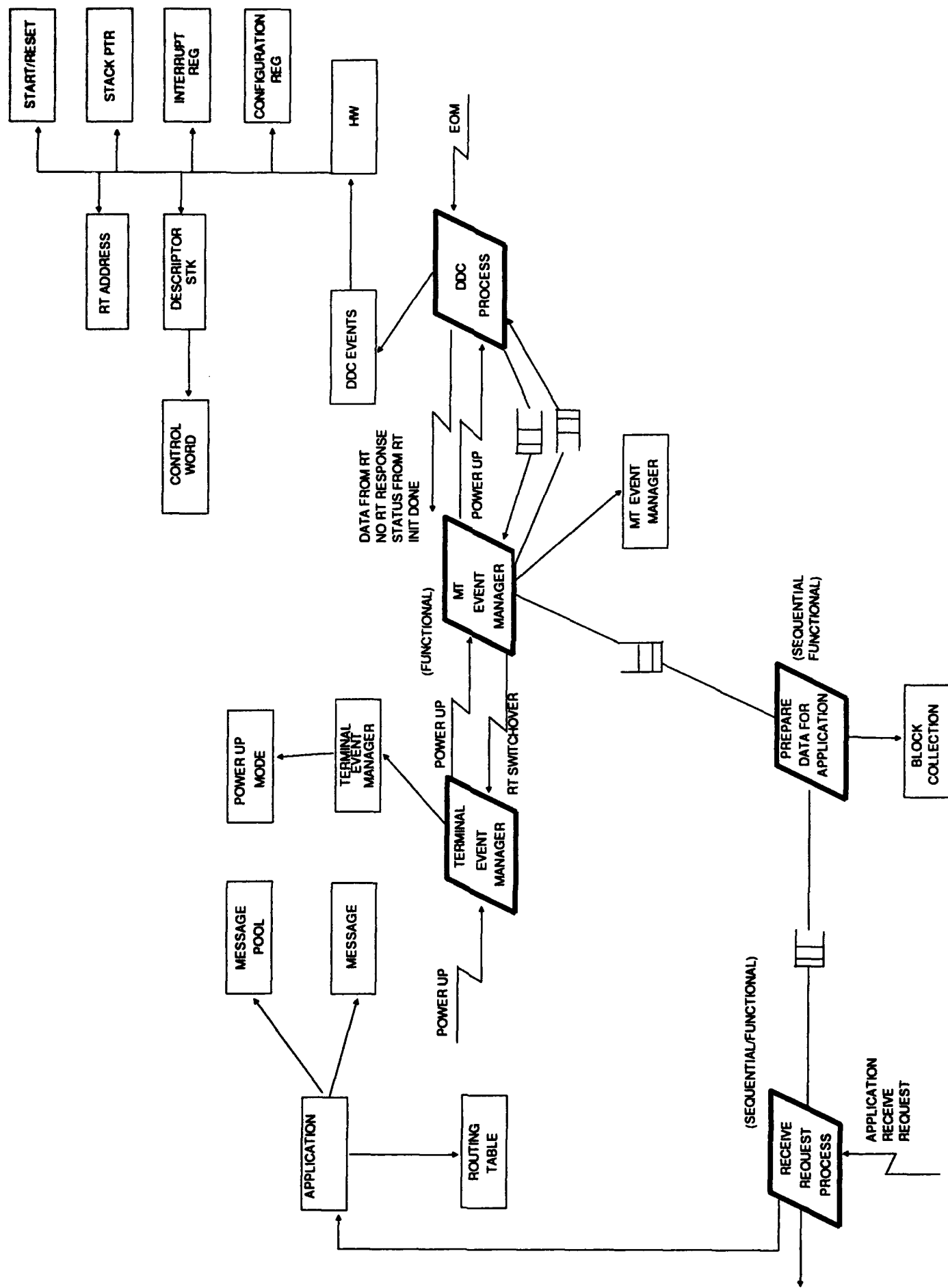
Update Console

# RT SOFTWARE ARCHITECTURE DIAGRAM



# MT SOFTWARE ARCHITECTURE DIAGRAM

50



## Terminal Information

1. Terminal Identifier	
2. Terminal Address	
3. Role	Bus Controller (BC)
4. Redundancy	Normal
5. Broadcast Address	0
6. Processor	Intel 80186
7. Ram	4
8. Rom	4
9. 1553B chip	1553B BCRT
10. Interface	Shared Memory
11. Diagnostic	>>>
12. Bus Assignment	>>>
13. Polling Sequence	0
14. Save Terminal	

Enter Selection:

***Questions or comments on content should be directed to:***

**K.C. King  
Boeing/STARS Demonstration Program Manager  
% Dual Inc.  
30 Skyline Drive  
Lake Mary, FL 32746  
(407) 333-8880**

**Jerri Turner-Harris  
Rockwell Command & Control Systems Division  
MS 460-220  
3200 East Renner Road  
Richardson, TX 75082-2402  
(214) 705-3151**

***Or to:***

**Grady Campbell  
Software Productivity Consortium  
2214 Rock Hill Road  
Herndon, VA 22070  
(703) 742-7104**

***Send feedback on the Consortium's Video Program and  
orders for video products to:***

**Technology Transfer Clearinghouse  
Software Productivity Consortium  
2214 Rock Hill Road  
Herndon, VA 22070  
(703) 742-7211**